Universität Augsburg

Dynamic ensemble forecasting of traffic flow by means of machine learning techniques

Matthias Sommer, Sven Tomforde

Report 2016-07 August 2016

Institut für Informatik
D-86135 Augsburg

Dynamic ensemble forecasting of traffic flow by means of machine learning techniques

Matthias Sommer* Sven Tomforde†

August 25, 2016

This article presents novel approaches to automatically learn the best combination of forecasts computed by several individual forecast methods. Ideas from the machine learning domain, such as Artificial Neural Networks and Learning Classifier Systems are adapted for this task. The combined forecast serves as basis for a pro-active adaptation of the control strategy in Organic Traffic Control (OTC). OTC is a decentralised, self-organised urban traffic control system that has the ability to optimise the signalisation, to establish progressive signal systems, and to offer route guidance recommendations. Besides analysing the success of the prediction strategy, we demonstrate the positive effect for OTC in terms of a simulation-based evaluation of an urban area situated in Hamburg, Germany. It reflects the actual topology, traffic data from a census, and the actual control strategy performed as reference. As a result, important figures such as the average waiting times at red lights and the emission values can be decreased significantly. Our findings support the hypothesis that the use of forecasts is beneficial for traffic control.

1 Introduction

The vehicular traffic domain is a vivid research field, both for industrial and academic research institutions. Novel trends, such as self-driving cars [Bir14], car-to-car communication [BBD+08], and traffic-adaptive control systems [SS10, STH16], have the goal to optimise the existing traffic infrastructure towards a more efficient utilisation of road networks. Still, we have to face negative impacts on the environment due to the increase

*Organic Computing Group, University of Augsburg, Germany
†Intelligent Embedded Systems Group, University of Kassel, Germany

of mobility, especially in urban areas. Consequently, this leads to an increase in pollution, a raising number of incidents, and an inefficient use of the transportation system. Urban road networks are characterised by a great number of signalised intersections in vicinity that need to be monitored and optimised. Due to the mathematical complexity, the online optimisation of a single signalised intersection is not feasible with analytical methods. Above all, the unpredictable behaviour of humans, and its highly complex dependencies between several streams throughout the road network make it an interesting and challenging field for self-organising solutions.

Currently installed traffic control systems (such as SCAT [SD80] and COMPASS [MW91]) usually rely on a centralised control centre, where all data is monitored and processed, and decisions about adaptations of the underlying strategy are made. Obviously, this results in a single point of failure, a high computational overhead, and a high demand of computational power. Thus, centralised solutions are not able to cope with expanding networks and future demands. Novel design trends lead to distributed, autonomous traffic control systems, such as InSync [SS10], and research projects, such as Organic Traffic Control (OTC) [PTB+11a]. Usually, real-world systems rely on fixed-time signalisation that was optimised manually by traffic engineers during design-time. Obviously, these static signal plans can not be optimal in every situation and are prone to obsolescence due to changing traffic conditions. On the contrary, an adaptive system, such as OTC is able to select the most appropriate phase durations at intersections based on the current traffic conditions and learns the impact of this situation-dependent selection in order to improve its behaviour over time while respecting safety corridors.

Current traffic-responsive control systems react on the currently observed conditions, typically measured in terms of traffic flows. This has several drawbacks, e.g., measuring traffic flows is always a trade-off between stable and recent trends, and the observed flow values describe the past conditions rather than those upcoming. Forecasts of future traffic conditions can be especially useful for route guidance [Fu01]. A variety of forecast techniques has been developed to predict traffic flow conditions [BF12]. Typically, these techniques are characterised by different strengths and weaknesses – meaning there is no such thing as an optimal technique. The novel mechanisms presented in this article introduce approaches to avoid a design-time decision about which technique to apply. Therefore, they automatically learn the most appropriate selection strategy of available forecast techniques for a certain traffic situation at runtime. As a result, the approaches take advantage of the different strengths without suffering due to the particular weaknesses of the different forecast methods.

Learning the best situation-to-method mapping replaces the problem of which technique to prefer by the questions of how to describe the particular situation and how to automatically learn this mapping. In order to demonstrate the potential benefit of our approaches, we analyse the performance from two different points of view. Initially, we show that the developed learning mechanisms outperform the individual techniques and other combination strategies. This is accompanied by evaluating the impact of predictions within the learning-based traffic control strategy of OTC. Therefore, we developed a simulation of an urban area situated in Hamburg, Germany that reflects the actual topology, traffic conditions as derived by a census, and the actual control strategies run-

ning in reality. Overall, we propose and evaluate a highly self-adaptive technique to relieve an engineer from complex design-time situated tasks related to forecast selection and configuration.

The remainder of this article is organised as follows. First, we introduce the OTC system and thereby give a brief overview to the basic capabilities traffic light control. Afterwards, we describe the state-of-the-art in time series forecasting. Section 4 explains the runtime learning approaches to combine and weight the results of various forecast techniques. The approaches are evaluated in Section 5 by applying simulations of a traffic network situated in Hamburg, Germany and with experiments based on real data from the Minnesota Department of Transportation. Finally, we summarise our findings and give an outlook to future work.

2 Organic Traffic Control

The Organic Computing (OC) initiative [MS04] postulates to master complexity in technical systems by equipping them with characteristics of natural – or *organic* – systems. On the one hand, this means to enable systems with capabilities of self-adaptation and self-optimisation of their runtime behaviour, and consequently with a robust self-organisation mechanism. On the other hand, this results in moving traditional design-time decisions to runtime and into the responsibilities of systems themselves. Due to the dynamic nature of vehicular traffic, control of traffic lights at intersections is an ideal testbed of OC principles – which resulted in the development of the OTC system [PTB+11b].

The architectural design of the system is illustrated by Figure 1. It follows the basic Observer/Controller design that is typical for OC systems [TPB+11]. The basic OTC system is responsible for adapting phase durations (i.e. green times of traffic lights at an intersection) according to the currently observed traffic conditions. Thereby, a safety-oriented, learning concept is applied to allow for a continuous self-optimisation process while simultaneously staying within controllable boundaries of system behaviour. In general, OTC establishes a multi-layered feedback loop that works on a rule-based learning technique and generates its behavioural strategies in an autonomous manner. OTC works in fully decentralised mode – meaning that one individual instance of OTC deployed on an intersection controller controls one intersection only.

Layer 0 of the architecture is a parametrisable fixed-time traffic light controller (TLC). Based on interfaces for the observation and configuration of Layer 0, the Observer/Controller structure of Layer 1 implements a rule-based adaptation process. The Observer analyses the current traffic situation and provides a situation description that is used as decision basis by the controller. The controller performs two tasks: 1) the success of the previous decisions is estimated (i.e. in terms of averaged delays) and the corresponding rules are updated; 2) the most promising rule matching the current situation is selected – resulting in a modification of the green times of the TLC at Layer 0. Due to safety reasons, Layer 1 only operates on existing rules and is restricted to exclusively using similar rules to the currently observed situation. In order to be able to react appropriately in case

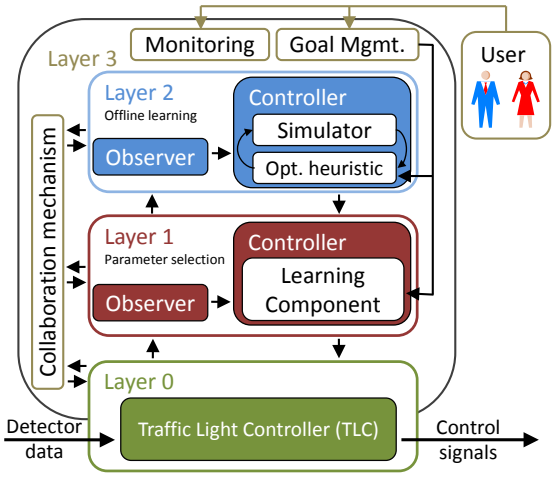

Figure 1: Multi-layered architecture of the OTC system.

of previously unknown situations, Layer 2 is responsible for generating novel knowledge. Therefore, a simulator (configured with the intersection's topology and the particular traffic situation) is coupled with an optimisation heuristic to find the best possible setup of green phase durations with regard to the particular traffic situation. Layer 3 provides the interface to users and administrators of the system. Further details on the learning mechanism can be found in [PRT+08].

3 State of the art in time series forecasting

In general, a time series is described by a time-ordered sequence of data points $X_1, ..., X_t$, derived from a system in discrete time intervals of successive measurements by sensors [AA13]. The data points can be seen as random variables, where X_1 denotes the measured value at the first point in time, X_2 the second value, and so on. A time series can have a trend (long-term increase or decrease), be cyclical (fluctuations with no fixed period), exhibit seasonal behaviour (seasonal factors with fixed period), or have irregular components (random component).

3.1 Methods for time series forecasting

Short-term traffic forecasting is an important aspect for *Intelligent Traffic Systems*. Typically, forecast techniques take the last monitored traffic flows to predict the estimated traffic flow conditions of the near future. There are also some techniques that rely on aggregated historical information (e.g. *Daily Load Curves* [CKWS04]) or a combination of both. Furthermore, some advanced forecast techniques allow to add more input variables, such as traffic density, current weather conditions, or data from adjacent roads (e.g. *Kalman filters* [GLW97] or *Artificial Neural Networks* [DC97]).

Parametric regression: Parametric regression is the search for the coefficients of a polynomial which minimises the sum of the quadratic errors for a set of sample data [Alp08]. Members of this class are typically working on past and current observations. Representatives are e.g. Moving Average, Exponential Smoothing, and Double Exponential Smoothing [Kal]. Only a small set of past traffic flows is needed to be able to respond to unknown situations. These models are easy to implement and fast to execute. A drawback is that their forecast accuracy tends to be lower than more advanced techniques such as the class of parametric regression models called ARIMA. ARIMA (Autoregressive Integrated Moving Average) is a statistical time series model, trying to fit the mathematical model underlying the time series [ZL02]. In contrast to other approaches, it is able to approximate non-stationary time series. The ARIMA model can be extended by a seasonal component (SARIMA), as traffic flows exhibit cyclic behaviour [SWO02]. Therefore, SARIMA is able to deal with non-stationary, seasonal data as well.

Non-parametric regression: In the domain of time series forecasting, non-parametric regression methods have several benefits over parametric methods [VFC07]. Härdle [Här90] gives an extensive introduction into the basics of non-parametric regression and compares representatives of the previously mentioned methods. He states that non-parametric approaches provide a versatile method to explore a relationship between two variables. They need no fixed parametric model to give forecasts and they deal well with missing values by interpolating between adjacent points. Non-parametric estimation methods are more flexible and well adaptable to local features, and multi-step forecasts are easily raised. However, they need more sample data than parametric regression methods and have higher computational costs. Representatives for this class are, e.g. K-nearest neighbor, Kernel methods (such as Support Vector Machines and Gaussian Process Regression) and spline smoothing.

Machine learning algorithms: A third class of techniques follows the concept of machine learning. Alpaydin [Alp08] defines the term machine learning as the optimisation of parameters describing a certain model. This model is optimised according to sample data or by past experiences measured based on an optimisation criteria. Bayesian Networks ([CMS08]) offer a powerful method for time series forecasting. These approaches are based on probabilistic graph models, where the nodes of a directed acyclic graph are random variables and the edges conditional dependencies. A more sophisticated machine learning approach is the Artificial Neural Networks (ANN): a highly simplified analogy of a biological neural system that is able to deal with uncertainties and chaotic data. As stated in [STH13, STH15b], ANNs are able to predict traffic flows in a reliable fashion. Here, multitask learning is used as it can improve the generalisation of the network as well as the forecast accuracy. Furthermore, ANNs offer the ability to add additional input, such as traffic density or the average speed of vehicles. Besides the mentioned approaches, several others, such as Kalman filters [OS84] and Support Vector Machines [Mar02] have been successfully applied to traffic flow forecasting.

3.2 Combining forecasts

In the following, we present approaches to combine the results of different techniques. A collective consideration of several forecast models is often expected to be a more powerful approach than just relying on one individual technique. A number of studies and reviews focus on the combination of several different methods [TWX+09, Arm01]. Several researchers investigated the strengths and weaknesses of linear combination strategies in comparison to individual forecast techniques [HE05, AA14]. Their results indicate that the forecast accuracies of the single forecast methods vary notably and that all combination methods significantly reduce the forecast error. Alpaydin [Alp08] proposes to use different learning algorithms, the use of different hyper-parameters (a parameter defining the configuration of the underlying parametric model), the fusion of sensor data, and the utilisation of different training sets. Armstrong [Arm01] suggests the use of at least five different forecasts methods, using data from several sources, and to assert higher weights to methods that performed better than others in the past. Further approaches focus on network-wide forecast models. They are neglected in this article, since forecasts are only considered for local intersection-wide control strategies.

In order to improve the reliability of the forecast and to overcome limitations and drawbacks of individual techniques, different approaches to combine the results of a set of forecast techniques to one aggregated result have been discussed in literature [TWX+09, Zha12]. The simplest approach is to calculate the simple average (SA) of all forecasts. It is easy to implement and does not require any estimation of weights. Although it offers suboptimal weights, its results are often superior to more sophisticated methods. Larrick and Soll [LS03] demonstrated that this could also lead to worse performance in some cases. Alternatively, the forecasts are considered with varying impact, weighted linearly according to their expected accuracy. Usually the value range of the weights is restricted to be between 0 and 1 and the sum of all weights should equal 1. This leads us to the problem of determining the optimal weights for each forecast technique.

Bates and Granger [BG69] proposed a method, called *Optimal Weights* that calculates the vector w of weights to minimise the error variance of the combination of n forecasts. In this case, it is supposed that there is no correlation between two different forecast errors. *Outperformance* is an approach introduced by Bunn [Bun89], where each weight is seen as the probability that the according forecast technique will perform best in the next time step. This is done by managing counters how often each technique performed best in the past few executions. Boosting and Bagging algorithms [AA13] try to obtain smaller forecast errors by combining multiple forecast methods where each technique has to be at least slightly better than random guessing. Initially, they were designed for ensemble learning in classification tasks, but were also adapted to regression problems [AI99]. One famous representative is AdaBoost [Sch13] which combines many weak predictors to achieve more accurate forecasts. The final output by AdaBoost is a weighted majority vote of the votes of each method. The weights are derived based on the forecast accuracy of each method on a training set.

4 Learning to forecast

The performance of combined traffic flow forecasts depends highly on the chosen weights. One possible approach to configure these weights appropriately is to find the best configuration offline, i.e. at design-time. Unfortunately, this approach is very time-consuming and determines settings that are only optimal in the sense of working best on average for all considered situations. Since there is no "best" forecast technique, there is also no best set of weights for all situations. Therefore, we argue that the assigned weights have to be adjusted dynamically at runtime, depending on the currently observed traffic conditions and the history of the current situation [STH14]. We propose three novel approaches for finding the best weights: The first approach is founding on the idea of Learning Classifier Systems, the second one is based on Artificial Neural Networks, and the third one works on the basis of daily load curves.

4.1 Time series forecast component

We present a forecasting component which is responsible for deriving forecasts within OTC running at each intersection. This *Forecast Module* is located in the *Observer* on Layer 1 (see Figure 2). In every time step, the current situation (i.e. the traffic flows) is retrieved from Layer 0 and serves as basis for the individual forecasts of the available forecast methods. First, the monitoring component pre-processes the data and then passes it on to the *Forecast Module* where it is stored in a time-ordered fashion. The forecasts of several forecast methods are accumulated into one comprehensive result and then combined with the current situation coming from the *Situation Analyzer*. Therefore, the situation does not only consider the current traffic behaviour, but also the respective forecasts. This new situation is then used for the selection of a matching signal plan in the *Controller*.

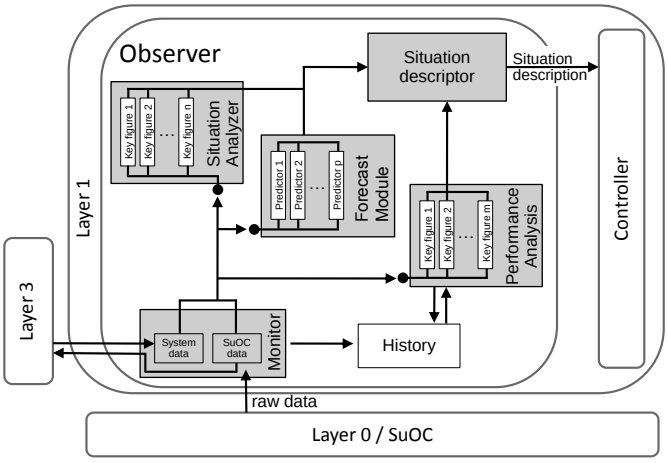

Figure 2: Close-up on Layer 1 of the architecture shown in Figure 1.

The schematic view of the forecast module is depicted in Figure 3. A `ForecastAdapter`

serves as interface for other modules that want to utilise its forecast ability. In case a module demands a forecast, the `ForecastModule` receives this request and informs all active forecast methods. These methods have to follow a certain abstract schema with defined input and output interfaces. Each forecast method then creates a forecast based on its local model and logic, and returns it to the `ForecastModule`. This set of forecasts is then passed on to the `ForecastCombiner` where it is combined based on the chosen `CombinationStrategy` into one comprehensive result. Again, the realisation of this combination is not fixed and can be implemented in a variety of ways (see Section 3.2). Finally, the combined result is returned to the module that demanded this forecast. Additionally, the `ForecastEvaluator` offers statistical measures to monitor the accuracy of the combined forecasts compared to the actual values. The `TimeseriesAnalyser` automatically classifies time series based on their characteristics (such as trend, seasonality or non-stationarity). If necessary, it processes a time series to make it stationary.

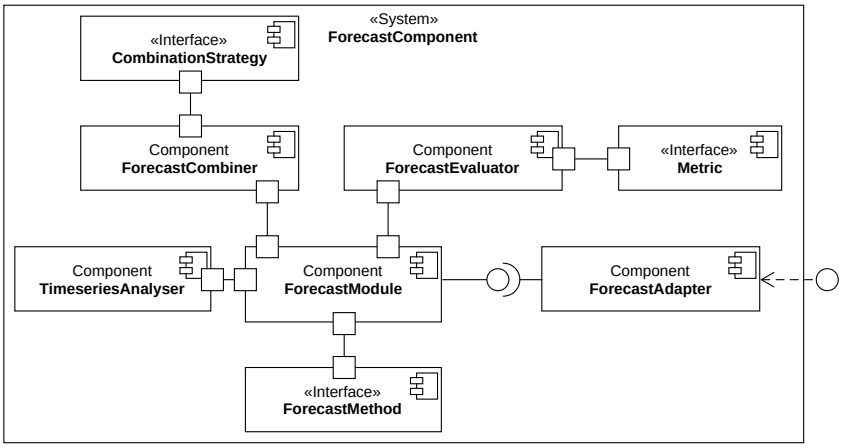

Figure 3: Schematic view of the forecast component.

4.2 Problem formulation

The goal of the forecast component is to automatically learn a function g with

$$g(sit) \rightarrow (w_1, \ldots, w_n) \tag{1}$$

The term *sit* defines the currently observed situation and w_1 to w_n the weights for the individual forecast techniques (with $n > 1$ and $w_1 + \cdots + w_n = 1$). Within each evaluation cycle of the OTC system (i.e. each loop when deciding about adapting the phase durations for the traffic lights), the weights are altered according to the estimated reward of g. In general, g is a function that maps situations to the configuration of the weights. This implies two basic tasks:

1. We need a description of the situation that serves as parameter for g, and

2. we need a (machine learning) technique that is able to improve this mapping over time with increasing experience.

In the remainder of this section, we introduce three different approaches to answer the second question. In addition, we analyse which approaches are possible to answer the first one (see Section 4.6).

4.3 Variant 1: ANN Historical Weighting

The first approach is called *ANN Historical Weighting*, which is based on Artificial Neural Networks (ANN) [Cru06]. ANNs have proven to be a powerful tool for classification tasks, gesture and hand writing recognition, forecasting of time series, and other problems [Gur97]. This makes them an appropriate method for the determination of the optimal vector of weights for a set of forecasts.

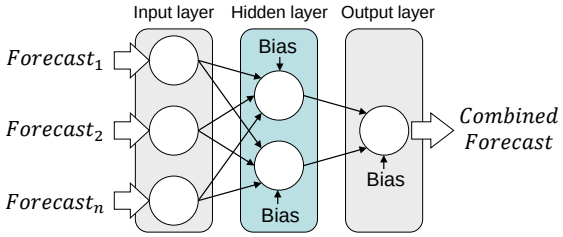

Figure 4: Combining forecasts with an ANN.

Figure 4 depicts an exemplary ANN. It consists of several neurons which are structured into several layers (input layer, hidden layer, and output layer). Each neuron is a set of input values and its associated weights and a function which calculates the sum of these weights and maps it to an output. The ANN receives the forecasts of all forecast techniques as input and returns the combined forecast as output. Therefore, the number of incoming neurons in the input layer has to be equal to the number of forecast techniques and the number of output neurons in the output layer is one. Usually, one hidden layer is used, whereas the number of neurons in that layer has to be determined through testing. Initially, a set of training data with an input (the forecasts) and the desired output value (the actual value) is necessary to train the network (to determine the best weights between the neurons). This is referred to as supervised learning. The training is done by a learning algorithm, such as Levenberg-Marquardt, Resilient Propagation or Backpropagation [HM94]. The possible topology ranges from Feedforward networks and Elman recurrent networks [Elm90] to Jordan recurrent networks [Jor86].

4.4 Variant 2: DLC Historical Weighting

The second approach, called *DLC Historical Weighting*, uses the basics of Daily Load Curves (DLC) to determine the best weights for the combination of forecasts. DLC categorises time series into several classes based on a distance metric (Figure 5). Time

series with similar patterns are therefore classified into the same class. Based on these classes, traffic flow can be forecasted as it shows typical patterns for work days, week ends, and holidays [CKWS04]. The idea to use DLC for the weight determination problem derives from the observation that each forecast technique offers similar forecast accuracies for similar situations. A situation is a time series of consecutive traffic flows over a defined time span. The similarity of two situations is determined by a distance measure such as the Manhattan Distance, Euclidan Distance, or Distance Time Warping [MÖ7]. The current traffic situation is compared to every known situation and the most equal one is chosen (in this case the one with the lowest Euclidean Distance). As the respective forecast error was stored for each situation, it can now be used to determine the weights for the current situation. The forecast error is converted into a weight by normalising its value to a range between 0 and 1. Therefore, forecast techniques that had a high forecast error, receive a low weight for the combination process and vice-versa. As more and more situations and forecast errors are stored during runtime, the system has the ability to learn which techniques to use in which situations and how to improve its forecast accuracy.

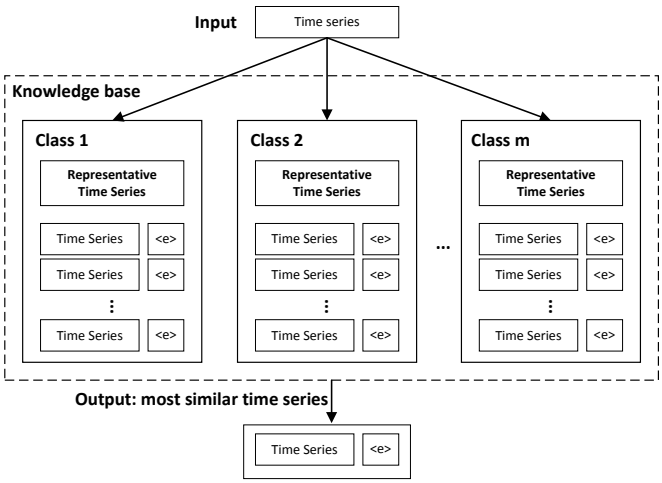

Figure 5: Classes of daily load curves mapping situations to weights.

Compared to the ANN-based approach, the DLC technique does not have to be trained during design-time as it is able to adapt to the behaviour of the used forecast techniques during runtime. However, the performance will be not as reliable as the ANN during the start-up period. Its accuracy also depends upon the configuration of the load curves, e.g. the maximum number of stored curves, their length, and the granularity of the measured data points describing each situation.

4.5 Variant 3: Extended Classifier Systems

Learning Classifier Systems (LCS) resemble a population-based machine learning technique combining ideas from evolutionary computing, reinforcement learning, supervised

or unsupervised learning, and heuristics [BK05]. LCS are adaptive, rule-based systems. The evolutionary algorithm is used to evolve these rules in order to search the problem space for the best solutions for a given optimisation task. The existing rules are rated upon their influence on the system or task. Therefore, the rules that have shown to achieve good results have a higher possibility to be chosen again in later executions.

Later, Lanzi et al. [LLWG07] proposed the *eXtended Classifier System for function approximation* (XCSF). XCSF tries to approximate functions in piecewise-linear fashion, such as $y = f(x)$ where x is the input and y is its payoff. A classifier is represented by 1) a condition defining the input, 2) an action to be executed, 3) a prediction of the estimated reward in case the action is executed, 4) a prediction error estimating the mean absolute deviation of this prediction, and 5) a fitness value which resembles the inverse function of the prediction error. Its basic process works as follows: First, the classifier system receives an input from its environment. Based on this situation description, matching classifiers are put into a match set. Because different classifiers can represent different actions, an action-selection-process has to be executed. This can either be done based on the fitness value of a classifier or as part of a fitness-weighted prediction average of all classifiers in the match set. All classifiers representing the chosen action are then put into a prediction array. The best action can be chosen deterministically (exploitation), randomly (exploration), or hybrid as a combination of both. Finally, the chosen action is executed and all classifiers in the prediction array are updated based on the received reward. Actions that improve the system performance gain a higher reward than others. Consequently, this allows the system to learn which actions are best to be performed in which situations. In case of observing a previously unknown situation, the evolutionary algorithm creates new classifiers. This can be done randomly or be a result of a guided process.

In our scenario, the input is represented by a vector of forecast values from several forecast models and the chosen action is the combination of these forecasts. Therefore, the vector contains as many entries as there are forecast methods. The action a resembles the combined forecasts. It is calculated as the fitness-weighted average as:

$$a = \frac{\sum_{cl \in P} pred_{cl_i} * f_{cl_i}}{\sum f_c l \in P} \tag{2}$$

with $pred_{cl_i}$ being the prediction value of the classifier i of the population P and f_{cl_i} being its fitness value. The reward is then calculated with a distance metric based on the combined forecast and the actual value. Here, we use the absolute value of the absolute forecast error. Most parameters of the XCS are set as suggested by Butz and Wilson [BW02]. Here, the population consist of a maximum of 800 classifiers. In case of reaching this limit, the least experienced classifiers are removed. Each classifier is represented as a rotating ellipsoid as described in Butz et al. [BLW08] which is assumed to cover the state space better than rectangles. Accordingly, the recursive least squares approximation was used for parameter estimation of the prediction values of the classifiers.

4.6 Situation encoding

A situation in Equation 1 represents the condition part in the learning function, while the particular weights for combining the forecasts define the action part. In general, different methods to determine the situation are possible:

1. The most simple approach to define the situation is to use the currently observed traffic flow condition as input. Hence, the observation of a traffic flow of $m\frac{vehicles}{hour}$ for the considered turning can be used to define *sit* in Equation 1. The advantage of this solution is the non-complex learning strategy and a feasible list of stored experiences. In contrast, the disadvantage is that neither the history that resulted in this traffic conditions nor the dynamics of traffic flows are considered.

2. Intuitively, using a list of the last x consecutive traffic flows provides a better basis than just the currently observed flows. This corresponds to the basic process model of a variety of forecast techniques: Using the last x observed traffic flows as input, the best configuration of weights is predicted instead of the next traffic flow. Compared to the first solution, a significantly larger number of situations has to be considered due to combinatorics – resulting in a condition space that grows as function of the number of historical flows that are taken into account. The disadvantages of the previous solution – i.e. neglecting the historical context and the dynamics of traffic – are suppressed, but still no classification of the general learning problem is achieved.

3. In order to derive more generalised classes of situations, the absolute values of the observations have to be replaced by an abstracted categorisation. Instead of working on absolute values, more generalised indicators are used to define the conditions. For instance, the percental change for the measuring points, an extrapolated trend, or statistical indicators of the forecast errors (i.e. standard deviation, variation, etc.) can serve as basis for defining *sit* in Equation 1.

We decided to start with an approach that is as close to the standard methods used for generating forecasts as possible. Therefore, we used the second approach for the XCS and ANN variants and defined the term *sit* as the last x consecutive traffic flows.

5 Evaluation

In the following, we present the results of two experiments. The first experiment was done with a freely available data set from the Minnesota Department of Transportation (MDoT)[1] which covers real-world data. We compare several forecast combination strategies and evaluate which strategy leads to the lowest forecast error. The second experiment is done within a simulation of an intersection in Hamburg, Germany. The intersection's simulation model reflects the real topology, traffic data from a census, and

[1] see http://www.fhwa.dot.gov/

the actual traffic light signalisation. The signalisation is optimised at runtime by the OTC system. This experiment's aim is to evaluate if the combination of forecasts and its application can lead to improvements of traffic control in a real scenario. We compare the results to a reference solution where the adaptation of the signalisation by OTC is done without forecasts.

5.1 Experimental setup

In accordance to the findings of Armstrong [Arm01] who suggested to combine 4 to 6 forecasts, we combine the results of 5 individual forecast methods: Moving Average (MA), Double Smoothing Average (DSA), Exponential Smoothing (ES), Double Exponential Smoothing (DES), and a Kalman Filter. In contrast to MA, where the result is calculated based on a SA of the previous observations, DSA, ES and DES assign various weights to those observations, whereas newer values are weighted higher than older ones. Therefore, the older the observation, the less important is its influence on the forecast. The Kalman Filter has to be trained first in a supervised fashion based on labelled training data before it can be applied to forecast time series.

The forecasts of these models were then combined with different combination strategies:

Optimal Weights takes the average of the last 3 forecast errors of each forecasting technique to determine the weights for the combination process. A small parameter test suggested this as the optimal setting. The covariance matrix contains only the individual forecast error variances and is therefore restricted to only be diagonal.

The *Outperformance* method estimates the probabilities of each forecast technique to perform best in the next execution based on the last 10 execution results. This parameter setting returned the best results in test runs with values of 5, 10, 15, 20, and 25 executions.

SA and the *Median* combination strategy do not need any parametrisation.

The *ANN* was trained with 5, 7, and 10 hidden neurons whereas 7 was chosen as the optimum number as it resulted in the lowest forecast error for the training sample and short training durations. We use 2 weeks of data (4032 data points) to train the *ANN* before it is ready to be executed. In this scenario, a simple Feedforward ANN with one hidden layer was used. The activation function for the input layer was the sigmoid function. The hidden and output layer were assigned with the linear activation function. The training was done with the Levenberg Marquardt algorithm as it proved to result in higher accuracy with lower execution time than Backpropagation and other second-order algorithms [HM94]. These design decisions are based on findings of an extensive parameter study presented by Sommer et al. [STH15b].

The *Historic Load Curves* strategy measures the similarity of two time series with the euclidean distance. For this evaluation, a maximum of 10 previous daily load curves per forecast method, with each curve consisting of the last 8 traffic flows, is stored. In case the storage is full, the oldest entry is removed. The similarity of two time series is measured by the Euclidean distance, and the time series with the lowest distance is chosen. For each forecast method, the forecast error from the most similar load curve

is taken as the weight for the combination of the current forecasts. The weights are then normalised to sum up to one and finally, the forecasts are combined with a simple weighted sum.

5.2 Scenario I

The evaluation of the first scenario was done with real-world traffic data from the MDoT which provides historic traffic data (i.e. information about traffic flow, detector occupancy, average density, and average speed) from the last 365 days, derived from detector stations along different highways, measured every 30 seconds. The data set used in the following experiments was measured by detectors at the Interstate 35E, in the south-west of Eagan and averaged over a 5 minute interval ranging from 2013/02/01 to 2014/01/31. This interval was chosen in accordance to the suggestion by Vlahogianni et al. [VGK04] as traffic flow and speed fluctuate too much for smaller intervals. Therefore, forecasts are generated accordingly for five minutes into the future. The training set consists of two weeks of data and the test set of the respective week afterwards. The maximum traffic flow was 2200 vehicles per hour. For every experimental run, the position of these sets is chosen randomly out of the whole data according to the principles of cross validation. The experimental results are averaged over 30 independent runs, showing the average absolute forecast errors, their standard deviations, and further outliers.

5.3 Evaluation results for Experiment 1

This experiment evaluates the models with cross-validation techniques for two weeks of training data. The results of this experiment can be seen in Figure 6. The box plot presents the statistical distribution of the average absolute forecast errors. The bottom and the top of each box represent the first and third quartiles, and the band inside the box shows the median. Outliers are indicated as separate points. The vertical axis shows the absolute forecast error in vehicles. Our findings can be summarised as follows: The average absolute forecast errors range between 57 and 71 vehicles over all combination techniques. ANN delivered the best results with an average error of 64.49, having a higher standard deviation. On the other hand, Outperformance gave the worst results with an average error of 66.77 and a standard deviation of 2.13. Interestingly, the SA strategy, which combines all forecast values equally weighted, results in an average performance with the overall lowest standard deviation of 1.36. It therefore offers a robust and reliable method.

Table 1 shows the average absolute forecast errors, calculated over 30 runs, their standard deviation, and the confidence interval with a level of confidence of 99% (significance level of 0.01). From the data in Table 1 it can be concluded that, although the ANN has the highest standard deviation, it offers the lowest lower (63.60) and the lowest upper bound (65.38), resulting in the overall best approach. Furthermore, the XCSF has the highest upper bound (67.66) of the confidence intervals.

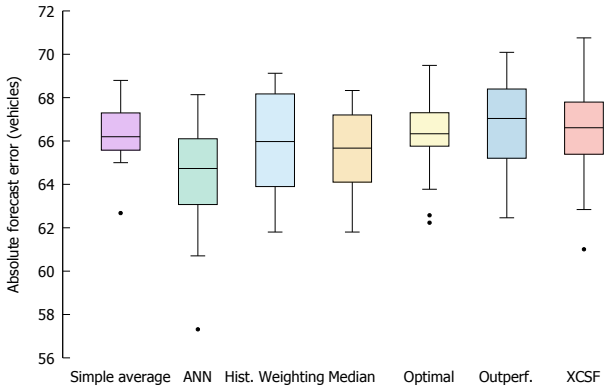

Figure 6: The absolute forecast error for all techniques averaged over 30 runs (lower values are better).

Table 1: Statistical measures for Experiment 1

Combination model	Mean error	Std. Dev.	Confid. int. (99%)
SA	66.46	1.36	[65.95, 66.97]
ANN Weighting	64.49	2.38	[63.60, 65.38]
XCSF	66.61	2.09	[65.55, 67.66]
Historical Weighting	65.88	2.28	[65.03, 66.73]
Optimal Weights	66.33	1.77	[65.67, 66.99]
Median	65.64	1.91	[64.93, 66.36]
Outperformance	66.77	2.13	[65.97, 67.56]

5.4 Evaluation results for Experiment 2

Our second experiment covers the whole data set containing one year of traffic flow data. Our hypothesis is that the machine learning algorithms ANN and XCSF need more data to reach their best performance. Therefore, a longer period of sample data is assumed to decrease the mean error of their forecast combination. The results are presented in Figure 7. The box plot shows the average absolute forecast errors in vehicles for all combination strategies and their respective standard deviations. The ANN delivered the combinations with the lowest forecast errors. In contrast to the previous experiment, Median and XCSF performed on an almost equal level, with ANN having a slightly lower standard deviation. The standard deviation of the forecasts errors ranges from 57 (Median and ANN) to 88 vehicles (Outperformance). The online learning methods, such as ANN and especially XCSF are supposed to improve their performance for longer training durations. To conclude, the ANN did not only deliver the most accurate combinations but offers a low standard deviation as well.

In order to evaluate how often a certain strategy performed better than another one,

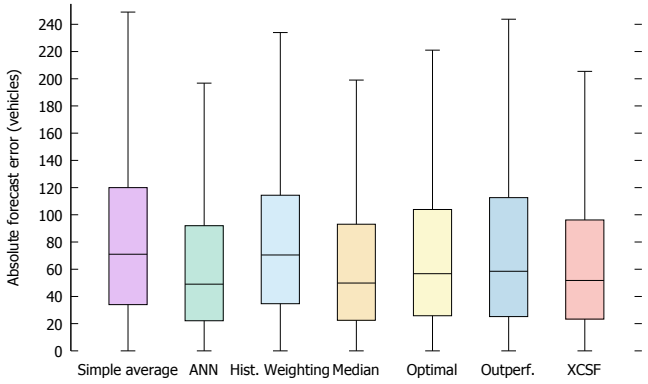

Figure 7: The absolute forecast error for all strategies for 1 year (lower values are better).

we considered the percentage better measure [Flo86] which is computed as follows:

$$PB_R = 100 * \frac{1}{n} \sum \delta_{i,t} \tag{3}$$

where

$$\delta_{i,t} = \begin{cases} 1 & \text{if } |e^R_{i,t}| < |e^S_{i,t}| \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

For every time step t, we evaluate which model i has the lowest forecast error $e_{i,t}$. According to Equation 3, this model receives 1 point. All other models get no points for this execution. Calculated over all time steps, the following results, shown in Table 2, were produced.

Table 2: Percentage better metric for Experiment 2

Combination model	Percentage better
All Equal	7.62 %
ANN Weighting	20.37 %
XCSF	10.34 %
Historical Weighting	7.81 %
Optimal Weights	20.21 %
Median	16.12 %
Outperformance	17.53 %
Sum	100 %

Our findings strongly support the view that ANNs are a suitable tool for the combination of forecasts. In accordance with Figure 6 and Figure 7, the ANN delivered the best combination results (20.37% of all executions) more often than other approaches. It has to be said that the ANN has previous knowledge as it was trained with sample data

beforehand. Although Optimal Weights showed only average performance regarding the forecast error, it came up with the best results in 20.21% of all executions. Interestingly, a simple approach, such as the median, had a percentage better of 16.12%. This suggest that the forecasts of the individual forecast methods do not tend to systematically under- or overestimate the actual value.

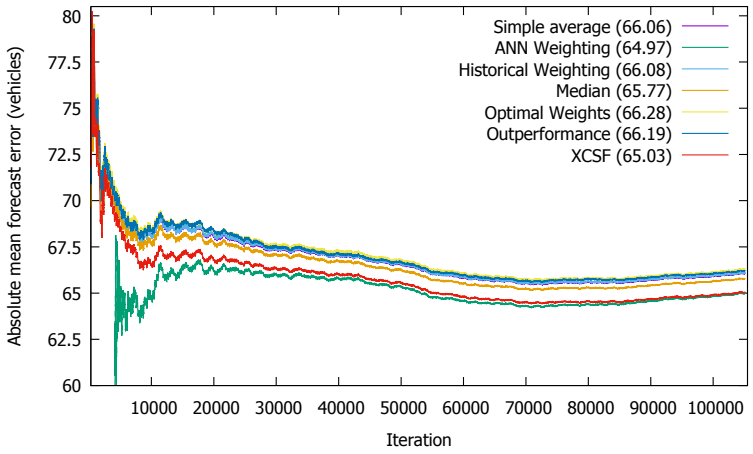

Figure 8: The average absolute forecast error curves over 1 year (lower values indicate better combination results).

Figure 8 presents the mean absolute forecast error curves for all combination models plotted over 105,000 iterations (resembles one year of data). The values given after the method names represent the average forecast error for this particular method. In the first 5,000 iterations, XCSF is still in its training phase, showing highly fluctuating results. Its initial classifiers assign equal weights to all forecasts and the system still tries to learn which classifiers perform good in which situations. As can be seen from the plot, after 2,000 iterations, XCSF clearly outperforms the other models. In contrast, the ANN has to wait until it has received enough data to build its training set. Its first computation is done around 5,000 time steps. Again, the ANN model outperformed the other methods (mean error of 64.97), closely followed by XCSF (65.03). Therefore, both machine learning techniques demonstrate sophisticatedly that they are able to adjust themselves and to learn how to combined several forecasts in different situations.

Our findings support the view that machine learning techniques are a suitable tool for the combination of forecasts. They have shown to successfully combine the forecasts of 5 individual models while outperforming other well-known combination strategies. The following experiment evaluates the potential benefits of these approaches in a traffic control system in terms of lower travel times and lower fuel consumption.

5.5 Scenario II: Forecast-based traffic control

The following evaluation was done with AIMSUN 8.0 using a highly realistic model of a four-armed intersection located at Hamburg, Germany (Figure 9) as part of the federal highway 433. The simulation model depicts the real topology including the actual fixed-time signal programs that have been developed by traffic engineers. Furthermore, the traffic data is based on census data performed by local authorities at Tuesday, May 4, 2004. In this census, cars and trucks passing the intersection were counted with a time resolution of 15 minutes. The simulation duration ranges from 5.30 a.m. to 12.30 p.m. therefore representing a typical morning rush hour scenario between around 6.45 a.m. to 10 a.m. Figure 10 illustrates the traffic profile throughout the simulation period. The horizontal axis shows the time, the vertical axis the traffic flow in *vehicles/hour*. It can be seen that the traffic density drastically raises up to 7,500 vehicles/hour between 6 a.m. and 6.45 a.m. After 9 a.m. the density starts to decrease with another minor peak at 10 a.m before it goes back to a normal level.

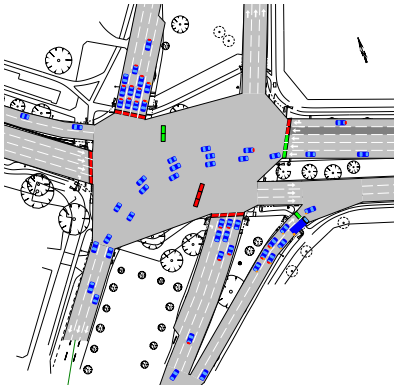

Figure 9: AIMSUN model of a four-armed intersection at Hamburg, Germany.

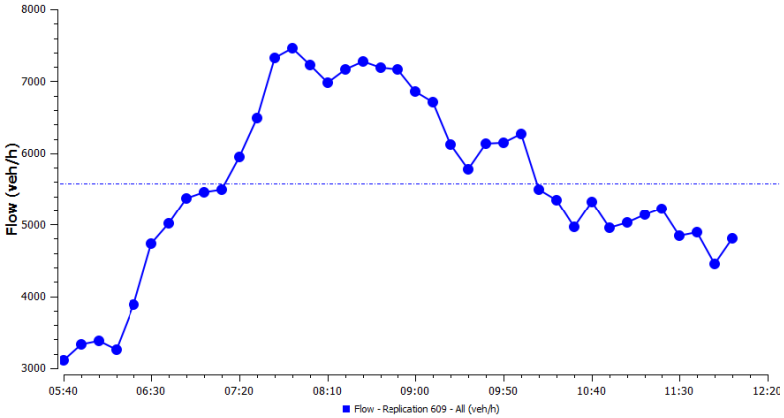

Figure 10: Vehicular traffic flow profile of a morning rush hour.

5.6 Evaluation results for scenario II

Table 3 which shows the average delay in seconds for each combination technique. The delay depicts the average time drivers need to get from an incoming section to an outgoing section of the intersection, including waiting times at red lights. OTC starts without further knowledge, only the standard signalisation is known in advance. Without the use of forecasts, OTC has an average delay of 163.2 seconds. Our study revealed that the highest improvement was achieved by SA (average delay of 145.0 seconds, 8.1% better than the reference run). On the contrary, the worst delays of the combination methods resulted with Optimal Weights and Median, still improving the results of the reference run by 2.1%. The use of forecasts seems quite promising as all combination strategies outperformed the reference run.

Table 3: Average delay in seconds for each combination strategy (lower is better).

Ref.	SA	Median	OP	OW	DLC	ANN	XCSF
163.2	**145.0**	159.7	159.7	154.4	155.8	153.6	156.0

Figure 11 presents similar results for the overall fuel consumption which was reduced from 1.63 to 1.59 litres consumed per vehicles. SA resulted in a higher consumption during the start-up period (the first hour of the simulation) where only few forecast algorithms are present (e.g. ANN need a warm-up phase). Afterwards, SA performs equally good or better than the reference run.

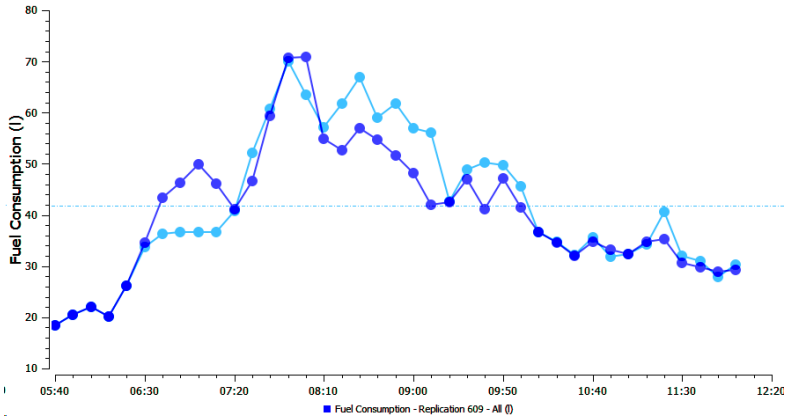

Figure 11: Comparison of the overall fuel consumption for the reference run (light blue) and the simple average (dark blue).

6 Conclusion

This article introduced novel learning approaches to combine different forecast techniques at runtime. Based on the motivation to pro-actively adapt the signalisation strategies

of the self-organised traffic control solution *OTC*, we developed three novel techniques for this learning task. One makes use of standard Artificial Neural Networks. It is trained in advance and then processed at runtime. The second technique resembles the concept of daily load curves by learning the situational context at runtime – i.e. it stores a sequence of the last traffic flow observations. Applying a distance metric to these time series allows us to chose the most similar situation and learn the corresponding weighting strategy. The last machine learning technique is based on Learning Classifier Systems. The eXtended Classifier System for function approximation was adapted to combine the results of several forecast methods. Our findings showed that this technique is able to automatically learn how to combine the forecasts in different situations.

We demonstrate the potential benefits of these approaches in two experiments. The first experiment makes use of freely available detector data from the Minnesota Department of Transportation. The novel learning approaches are compared to other combination strategies. The first scenario showed that these methods are capable of improving the overall forecast accuracy, with ANN offering the lowest forecast errors. After their warm-up time, XCSF and ANN outperform the other combination strategies.

Afterwards, we integrated the mechanisms in OTC and investigated the potential benefit for the traffic signal adaptation. The second scenario with a simulated traffic control system resulted in a slightly different outcome. Here, SA performed best. Our findings strongly support the view that the use of forecasts lowers the average travel times. All combination strategies lead to lower travel times compared to the reference run. To sum it up, ANN is a fast and reliable method to combine several forecasts offering convincing results. The DLC method on the other hand does prove that the use of forecasts has benefits, but one has to take compromises, e.g. the number of load curves and the granularity of the situation description have to be considered to hold down the computational effort. XCSF needs warm-up time to fill its knowledge base.

On this basis it can be concluded that combined forecasts, independently of the strategy, leads to a better control strategy, i.e. in terms of decreased travel times and fuel consumption.

Current and future work deals with a better integration of the developed forecast techniques into the OTC system. Besides a pro-active adaptation of the signalisation strategy [SH16], two important tasks have been identified in this article. On the one hand, we can use forecasts to generate new classifiers for novel situations in advance to their appearance. This means to trigger the Layer 2-based rule-generation component if the predicted traffic condition is either not matched by the current rule-set or all matching rules have proven to be not successful. On the other hand, forecasts of traffic flow conditions will be integrated into the routing mechanism of OTC [STH15a]. In contrast to considering just the observed situations being present in the overall network and basing the route recommendations on this information, a time-dependent approach is investigated.

References

[AA13] Ratnadip Adhikari and R. K. Agrawal. An introductory study on time series modeling and forecasting. *Computing Research Repository*, abs/1302.6613, 2013.

[AA14] Ratnadip Adhikari and R. K. Agrawal. Performance evaluation of weights selection schemes for linear combination of multiple forecasts. *Artif. Intell. Rev.*, pages 529–548, 2014.

[AI99] Ran Avnimelech and Nathan Intrator. Boosting regression estimators. *Neural Computation*, 11:491–513, 1999.

[Alp08] E. Alpaydın. *Maschinelles Lernen*. Oldenbourg, 2008.

[Arm01] J.S. Armstrong. *Principles of Forecasting: A Handbook for Researchers and Practitioners*. Int. Series in Operations Research & Management Science. Springer, 2001.

[BBD+08] Roberto Baldessari, Bert Bödekker, Matthias Deegener, Andreas Festag, Walter Franz, C. Christopher Kellum, Timo Kosch, Andras Kovacs, Massimiliano Lenardi, Cornelius Menig, Timo Peichl, Dieter Röckl, Matthias Seeberger, Markus Straßberger, Hannes Stratil, Hans-Jörg Vögel, Benjamin Weyl, and Wenhui Zhang. Car-2-car communication consortium - manifesto. Technical report, Car-2-Car Communication Consortium, 04 2008.

[BF12] E. Bolshinsky and R. Freidman. Traffic flow forecast survey. Technical report, Technion - Israel Institute of Technology, 2012.

[BG69] J. M. Bates and C. W. J. Granger. The combination of forecasts. *OR*, 20(4):451–468, 1969.

[Bir14] Michelle Birdsall. Google and ITE: The road ahead for self-driving cars. *ITE Journal*, 84(5):36–39, 5 2014.

[BK05] L. Bull and A. Kovacs. *Foundations of Learning Classifier Systems*, volume 183, chapter Foundations of Learning Classifier Systems: An Introduction, pages 1–17. Springer Berlin Heidelberg, 2005.

[BLW08] Martin V. Butz, Pier Luca Lanzi, and Stewart W. Wilson. Function approximation with XCS: hyperellipsoidal conditions, recursive least squares, and compaction. *IEEE Trans. Evolutionary Computation*, 12(3):355–376, 2008.

[Bun89] Derek Bunn. Forecasting with more than one model. *Journal of Forecasting*, 8(3):161–166, 1989.

[BW02] Martin Butz and Steward Wilson. An algorithmic description of XCS. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 6:144 – 153, 2002.

[CKWS04] R. Chrobok, O. Kaumann, J. Wahle, and M. Schreckenberg. Different methods of traffic forecast based on real data. *European Journal of Operational Research*, 155(3):558–568, June 2004.

[CMS08] E. Castillo, J. Menendez, and S. Sanchezcambronero. Predicting traffic flow using Bayesian networks. *Transport. Research Part B: Methodological*, 42(5):482–509, June 2008.

[Cru06] Holk Cruse. *Neural Networks as Cybernetic Systems*. 2006.

[DC97] Mark S. Dougherty and Mark R. Cobbett. Short-term inter-urban traffic forecasts using neural networks. *International Journal of Forecasting*, 13(1):21 – 31, 1997.

[Elm90] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

[Flo86] Benito E. Flores. Use of the sign test to supplement the percentage better statistic. *International Journal of Forecasting*, 2(4):477–489, 1986.

[Fu01] Liping Fu. An adaptive routing algorithm for in-vehicle route guidance systems with real-time information. *Transp. Research Part B: Methodological*, 35(8):749–765, 2001.

[GLW97] S. Garside, K. Lindveld, and J. Whittaker. Tracking and predicting a network traffic process. *International Journal of Forecasting*, 13:51–61, 1997.

[Gur97] Kevin Gurney. *An Introduction to Neural Networks*. CRC Press, 1997.

[Här90] W. Härdle. *Applied Nonparametric Regression*. Econometric Society Monographs. Cambridge University Press, 1990.

[HE05] Michèle Hibon and Theodoros Evgeniou. To combine or not to combine: selecting among forecasts and their combinations. *Intern. Journal of Forecasting*, 21(1):15–24, 00 2005.

[HM94] M. T. Hagan and M. B. Menhaj. Training feedforward networks with the marquardt algorithm. *Trans. Neur. Netw.*, 5(6):989–993, November 1994.

[Jor86] M. I. Jordan. Serial order: A parallel distributed processing approach. Technical Report ICS Report 8604, Institute for Cognitive Science, University of California, 1986.

[Kal] Prajakta S. Kalekar. Time series forecasting using holt-winters exponential smoothing.

[LLWG07] Pier Luca Lanzi, Daniele Loiacono, Stewart W. Wilson, and David E. Goldberg. Generalization in the xcsf classifier system: Analysis, improvement, and extension. *Evolutionary Computation*, 15(2):133–168, 2007.

[LS03] R. Larrick and J. Soll. Intuitions About Combining Opinions: Misappreciation of the Averaging Principle. Working paper, INSEAD, 2003.

[MÖ07] Meinard Müller. *Information Retrieval for Music and Motion*, chapter Dynamic Time Warping, pages 69–84. Springer, 2007.

[Mar02] Mario Martin. On-line support vector machine regression. In *Proceedings of the 13th European Conference on Machine Learning*, pages 282–294, 2002.

[MS04] Christian Müller-Schloer. Organic Computing: On the feasibility of controlled emergence. In *CODES and ISSS 2004 Proceedings*, pages 2–5. ACM Press., 2004.

[MW91] Lam K. Masters, H. and K. Wong. Incident detection algorithms for compass: An advanced traffic management system. In *Proc. of vehicle navigation & information systems conference*, volume 2 of *Vehicle Navigation and Information Systems*, pages 295–310, 1991.

[OS84] Iwao Okutani and Yorgos J. Stephanedes. Dynamic prediction of traffic volume through Kalman filtering theory. *Transp. Research Part B: Methodological*, 18(1):1–11, 1984.

[PRT+08] Holger Prothmann, Fabian Rochner, Sven Tomforde, Jürgen Branke, Christian Müller-Schloer, and Hartmut Schmeck. Organic control of traffic lights. In *Proceedings of the 5th International Conference on Autonomic and Trusted Computing (ATC-08)*, volume 5060 of *LNCS*, pages 219–233. Springer Verlag, 2008.

[PTB+11a] H. Prothmann, S. Tomforde, J. Branke, J. Hähner, C. Müller-Schloer, and H. Schmeck. Organic Traffic Control. In *Organic Computing – A Paradigm Shift for Complex Systems*, chapter 5.1, pages 431–446. Birkhäuser Verlag, 2011.

[PTB+11b] Holger Prothmann, Sven Tomforde, Jürgen Branke, Jörg Hähner, Christian Müller-Schloer, and Hartmut Schmeck. Organic Traffic Control. In *Organic Computing – A Paradigm Shift for Complex Systems*, pages 431 – 446. Birkhäuser Verlag, Basel, CH, 2011.

[Sch13] RobertE. Schapire. Explaining adaboost. In *Empirical Inference*, pages 37–52. Springer Berlin Heidelberg, 2013.

[SD80] Arthur G. Sims and Kenneth W. Dobinson. The Sydney coordinated adaptive traffic (SCAT) system – Philosophy and benefits. *IEEE Transactions on Vehicular Technology*, 29(2):130–137, 1980.

[SH16] Matthias Sommer and Jörg Hähner. Anticipatory adaptation of signalisation based on traffic flow forecasts within a self-organised traffic control system. In *Proc. of 5th International Conference on Transportation and Traffic Engineering (ICTTE 2016)*, 2016.

[SS10] Matt Selinger and Luke Schmidt. Adaptive traffic control systems in the united states: Updated summary and comparison. Technical report, HDR Engineering, Inc., 09 2010.

[STH13] Matthias Sommer, Sven Tomforde, and Jörg Hähner. Using a neural network for forecasting in an organic traffic control management system. In *International Workshop on Embedded Self-Organizing Systems (ESOS '13)*, 2013.

[STH14] Matthias Sommer, Sven Tomforde, and Jörg Hähner. Learning to Predict: Automated Management and Correction of Prediction Techniques for Traffic Flows within a Self-organised Traffic Control System. In *In. Proceedings of the 11th International Congress on Advances in Civil Engineering (ACE14), held 21-25 October, 2014 in Istanbul, Turkey*, pages 1–6, 2014.

[STH15a] Matthias Sommer, Sven Tomforde, and Jörg Hähner. Forecast-based route recommendations in organic traffic control. In *Proceedings of the 28th GI/ITG International Conference on Architecture of Computing Systems - ARCS Workshops*, 2015.

[STH15b] Matthias Sommer, Sven Tomforde, and Jörg Hähner. A systematic study on forecasting of traffic flows with artificial neural networks. In *3rd International Workshop on "Self-optimisation in Organic and Autonomic Computing Systems" (SAOS15)*, 2015.

[STH16] Matthias Sommer, Sven Tomforde, and Jörg Hähner. *Autonomic Road Transport Support Systems*, chapter An Organic Computing Approach to Resilient Traffic Management, pages 113–130. Springer International Publishing, Cham, 2016.

[SWO02] Brian L Smith, Billy M Williams, and R Keith Oswald. Comparison of parametric and nonparametric models for traffic flow forecasting. *Transportation Research Part C: Emerging Technologies*, 10(4):303–321, 2002.

[TPB+11] Sven Tomforde, Holger Prothmann, Jürgen Branke, Jörg Hähner, Moez Mnif, Christian Müller-Schloer, Urban Richter, and Hartmut Schmeck. Observation and Control of Organic Systems. In *Organic Computing - A Paradigm Shift for Complex Systems*, pages 325 – 338. Birkhäuser Verlag, 2011.

[TWX+09] M. C. Tan, S. C. Wong, J. Xu, Z. Guan, and P. Zhang. An Aggregation Approach to Short-term Traffic Flow Prediction. *IEEE Transactions Intelligent Transportation Systems*, 10(1):60 – 69, 2009.

[VFC07] J. M. Vilar-Fernandez and R. Cao. Nonparametric forecasting in time series. a comparative study. *Communications in Statistics - Simulation and Computation*, 36(2):311–334, 2007.

[VGK04] Eleni I. Vlahogianni, John C. Golias, and Matthew G. Karlaftis. Short-term traffic forecasting: Overview of objectives and methods. *Transport Reviews*, 24:533–557, 2004.

[Wil95] Stewart W. Wilson. Classifier fitness based on accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.

[Zha12] Yang Zhang. How to provide accurate and robust traffic forecasts practically? In *Intelligent Transportation Systems*, pages 189–206. InTech, 2012. ISBN 978-953-51-0347-9.

[ZL02] Yang Zhang and Yuncai Liu. Comparison of parametric and nonparametric techniques for non-peak traffic forecasting. *Transportation Research Part C*, 10:303–321, 2002.